第5章

Cベース設計を体験する

ImpulseC/Developerの活用事例

仲野 巧

米国Impulse Accelarated Technologies社のハードウェア/FPGA設計ツール「ImpulseC/Developer」は，C言語の記述からハードウェアのHDLを生成できるツールである．ただし，C言語でソフトウェアを記述するかのように設計するのではなくて，ハードウェアを考えてC言語で記述する意識が必要である．ここでは，C言語によるハードウェア開発の実際を紹介する．本誌2008年2月号付属DVD-ROMに収録の評価版を使えば，体験してみることもできる．（筆者）

最近，FPGA（Field Programmable Gate Array）を利用してハードウェア回路の実装を行う機会が増えています．このためハードウェアの知識がある技術者がハードウェアのイメージをC言語で記述し，実装から評価までの作業がスムーズになれば，効率良く開発が行えるようになります（図1）．

そこで本稿では，C言語から生成したハードウェア（VHDLコード）をFPGA向けツールでシミュレーションします（図2）．

1. 動作合成の概要

● 動作合成のパラメータ

CoDeveloperのハードウェア生成は，動作合成の指定（#pragma）によって性能（動作）が変わってきます．パラメータは，

```
#pragma CO
```

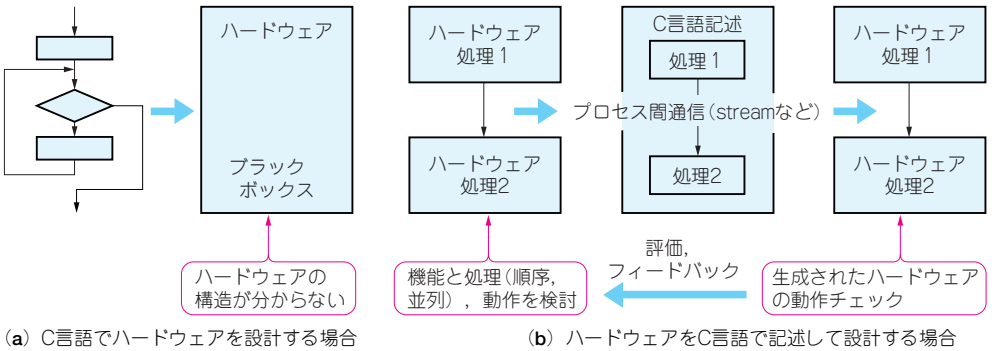
で指定します．これは，C言語のブロックの終わりを示すブラケット（}）までに適応されます．

代表的な指定を表1に示します．

● プロセス間通信

ImpulseC/CoDeveloperでは，処理をプロセス構造で設計します．データの入出力は，プロセス間通信で行います．

プロセス間通信は，図3のようにストリーム（Stream），レジスタ（Register），シグナル（Signal）などのチャネルがあります．データ処理にはFIFO（First-in First-out）メモリを利用したストリームを利用します．



（a）C言語でハードウェアを設計する場合　（b）ハードウェアをC言語で記述して設計する場合

図1
ハードウェアを考えてC言語で記述する意識が必要
「C言語でハードウェアを設計する」ことと「ハードウェアをC言語で記述する」ことは違う．

Keyword　ImpulseC/Developer，動作合成，プロセス間通信，ストリーム，平方根

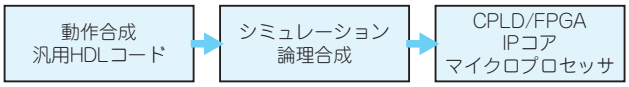


図2　動作合成による開発フロー

動作合成ツールによるHDLコードの生成からシミュレーション，論理合成，実装までのフローを示す．

表1　動作合成のパラメータ

パラメータ	説　明
#pragma CO UNROLL	UNROLLは，forループなどの繰り返しをループの回数だけ展開し，ハードウェアをコピーして生成する．従って，繰り返しの回数は，定数であることが必要で，ハードウェアの規模と速度を考慮して指定する必要がある．
#pragma CO PRIMITIVE	PRIMITIVEは，ハードウェア・プロセスから呼ぶことができるハードウェア・プリミティブ関数を生成する．プリミティブ関数は，呼び出されるごとにインスタンスが生成される．
#pragma CO FLATTEN	FLATTENは，switchや複雑なif-then-elseを平坦化したハードウェアを生成する．平坦化された回路は，多重ステージが1ステージで動作する．
#pragma CO PIPELINE	PIPELINEは，ループとパイプラインする行の前に指定することでパイプライン化されたハードウェアを生成する．パイプライン化は，値を保持するパイプライン・レジスタが挿入されるため並行して高速に動作する一方，ゲート規模が大きくなる．パイプラインの段数は，StageDelayで制御することが可能．
#pragma CO SET StageDelay 32	StageDelayは，パイプライン・ステージの最大値を指定する．

ハードウェアのプロセス・コンポーネントに接続するストリームのブロックを図4に示します．

●ストリームの信号名

標準（デフォルト）のストリームでは，リスト1に示すように，8ビットの値（datawidth）を4ビット・アドレスの16個（addrwidth）に入力と出力を行います．ストリームの信号は，入力の許可（en），レディ（rdy），終了（eos），データ（deta）と出力の許可（en），レディ（rdy），終了（eos），データ（deta）で制御します．

●ストリーム入力

ストリーム入力は，リスト2に示すように，en＝0，eos＝0でrdyを待ち，1になったらdataにデータを書き込み，en＝1でclkの立ち上がりエッジを待ち，en＝0で次のデータを繰り返し，最後にeosを1にします．

●ストリーム出力

ストリーム出力は，リスト3に示すように，en＝0で

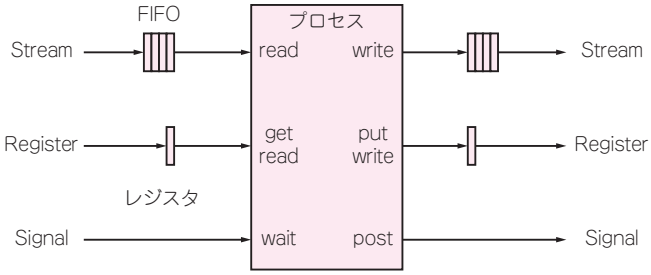


図3　プロセス間通信の種類

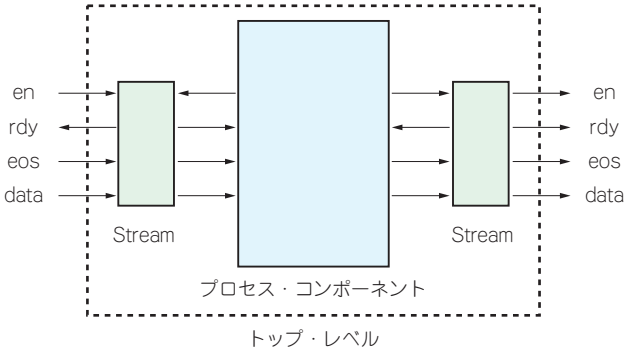


図4　ストリームのインターフェース

リスト1　Streamの信号名

```
entity stream is
  generic (
    datawidth : positive := 8;
    addrwidth : positive := 4
  );
  port (
    reset, clk : in std_ulogic;
    input_en : in std_ulogic;
    input_rdy : out std_ulogic;
    input_eos : in std_ulogic;
    input_data : in std_ulogic_vector (datawidth-1
                                        downto 0);
    output_en : in std_ulogic;
    output_rdy : out std_ulogic;
    output_eos : out std_ulogic;
    output_data : out std_ulogic_vector (datawidth-1
                                         downto 0)
  );
end;
```

リスト2　Streame入力

```
p_des_producer_blocks_out_en <= '0';
while (not endfile(text_file)) loop
    wait until p_des_producer_blocks_out_rdy = '1';
    read(text_file, ch);  -- Get character from file
    byte :=
          conv_std_logic_vector(character'pos(ch),8);
    p_des_producer_blocks_out_data <=
                          std_ulogic_vector(byte);
    p_des_producer_blocks_out_en <= '1';
    wait until rising_edge(clk);
    p_des_producer_blocks_out_en <= '0';
end loop;
wait until p_des_producer_blocks_out_rdy = '1';
p_des_producer_blocks_out_eos <= '1';
p_des_producer_blocks_out_en <= '1';
wait until rising_edge(clk);
p_des_producer_blocks_out_en <= '0';
```